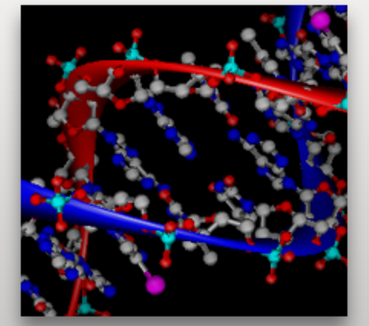
Building 64-bit Solutions for Tiger

Living in a double wide

Session 507

These are confidential sessions—please refrain from filming or taking pictures.



OS Foundations

Building 64-bit Solutions for Tiger

Living in a double wide

Stan Shebs
Software Engineer

http://developer.apple.com/wwdc2005/

Agenda

- What do we mean by 64-bit?
- What's new in Tiger for 64-bit applications?
- When should you create a 64-bit program?
- What to consider when writing a 64-bit program
- How to provide a GUI for your 64-bit program

Intel and 64-bit

?

What Is “64-bit OS Support?”

- The ability to do 64-bit integer math natively
- Support for more than 4GB of physical memory
- Support for virtual address spaces larger than 4GB

Mac OS X 64-bit Architecture

- Existing 32-bit programs unaffected
- No mixed 32 / 64-bit tasks
  - 64-bit program must have 64-bit version of each library used
- Universal binaries
  - For packaging 32-bit and 64-bit programs and libraries together
  - With Intel code included, binaries are “three-way”
- Single (32-bit) kernel for all Apple hardware
  - No rebooting necessary
  - Existing drivers continue to work

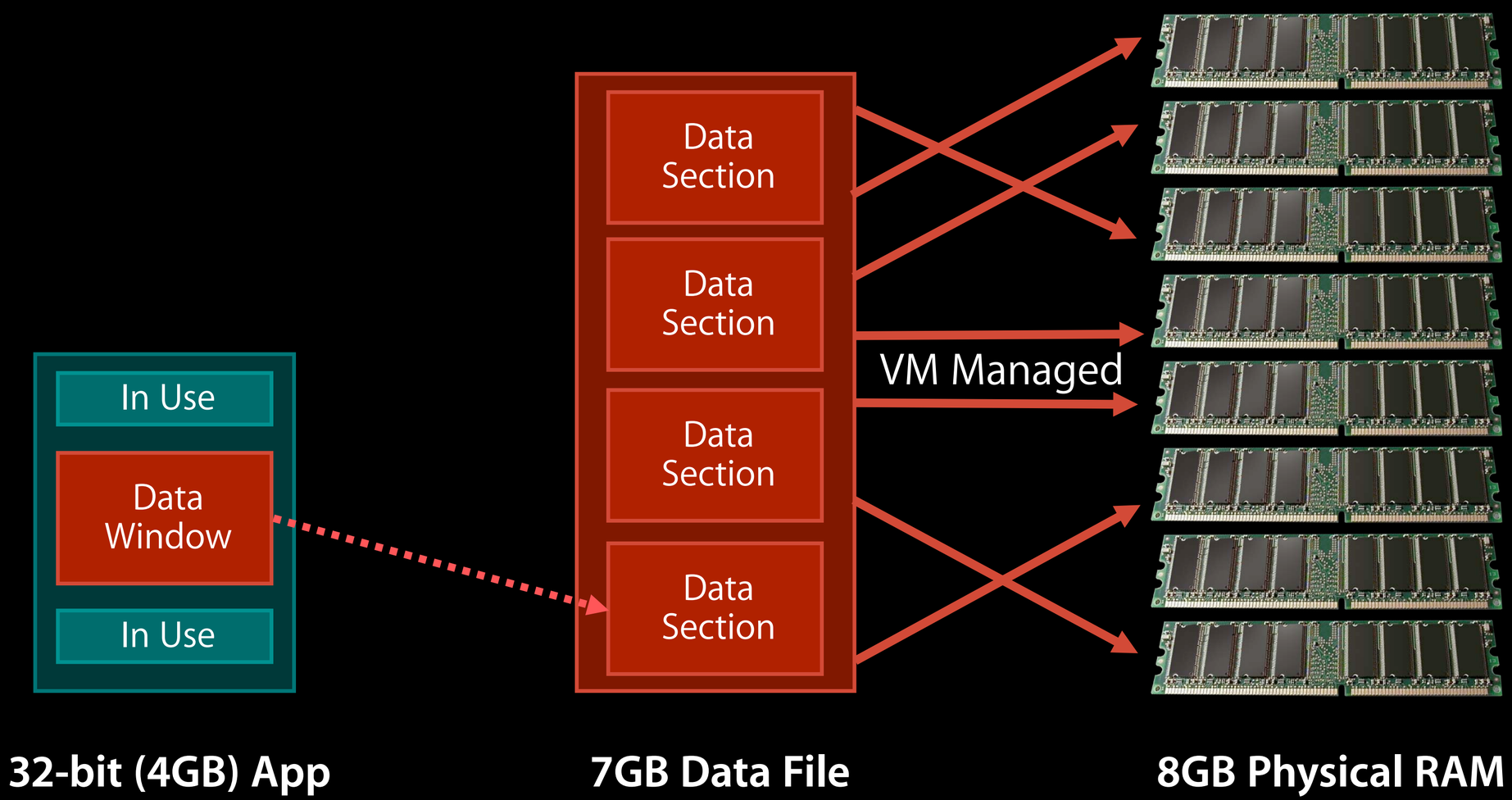
Panther First Brought 64-bit to the Mac

Okay, technically 10.2.7, but close enough…

- 64-bit instructions and data
  - Available to all programs
  - Some restrictions (leaf functions)
- Greater than 4GB physical memory support
  - Can cache large application data sets in memory
  - Have to “work” to access them
- G5-only

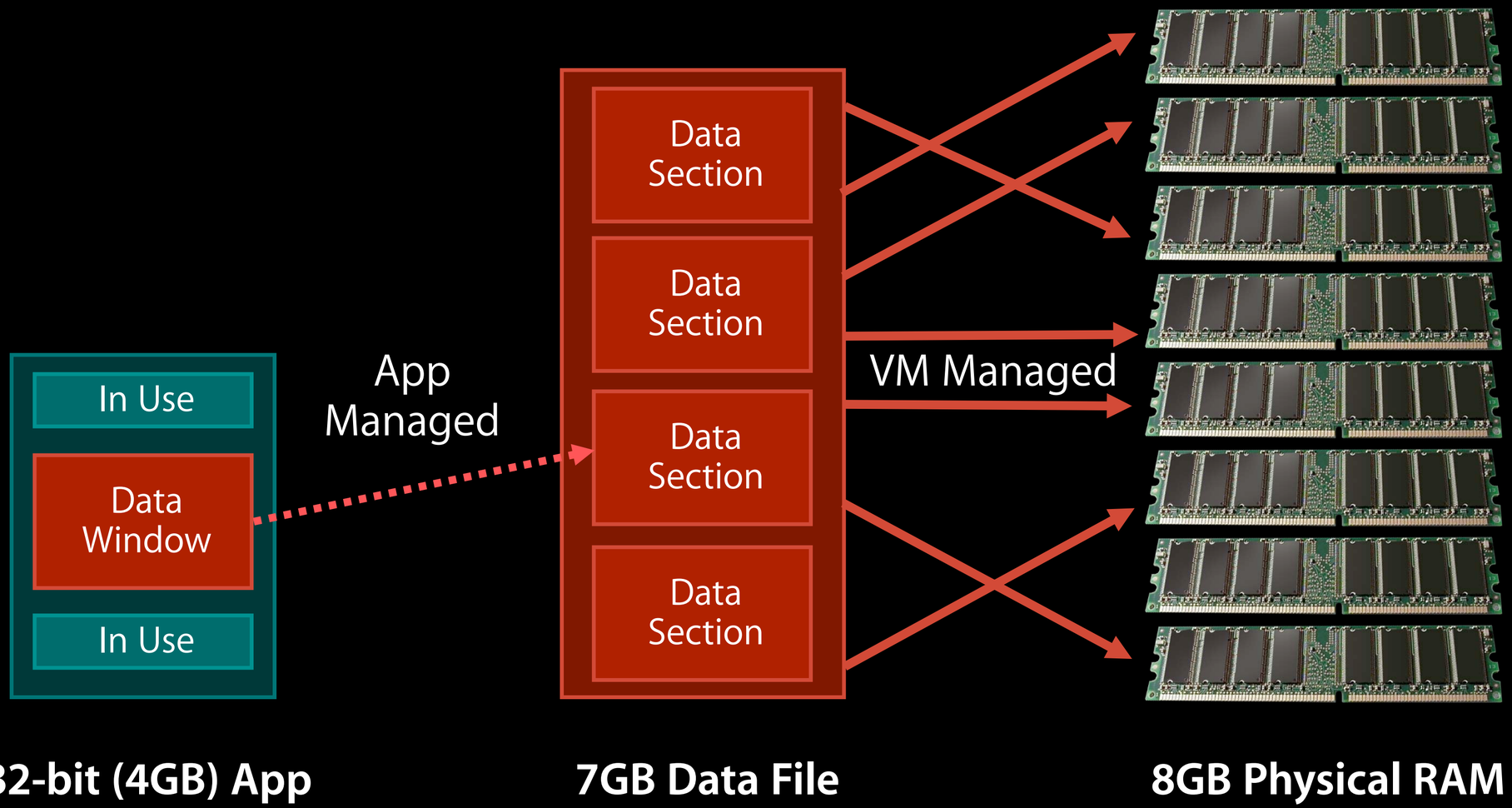
Large Data Sets in 32-bit Applications

A window for your single-wide



Large Data Sets in 32-bit Applications

A window for your single-wide



Tiger Adds to the 64-bit Mac Experience

- New 64-bit PowerPC ABI
  - Take full advantage of G5's 64-bit registers and instruction set
  - In all code—not just leaf functions
- 64-bit version of Mach-O file format
- Gigantic address space support
  - Allocate a single block of 2^49 bytes (512 terabytes) if you like
  - "Convenient" access to large cached data sets
- Still G5-only

How Gigantic Is a 64-bit Address Space?

16 exabytes

≈16 billion billion bytes

How Gigantic Is a 64-bit Address Space?

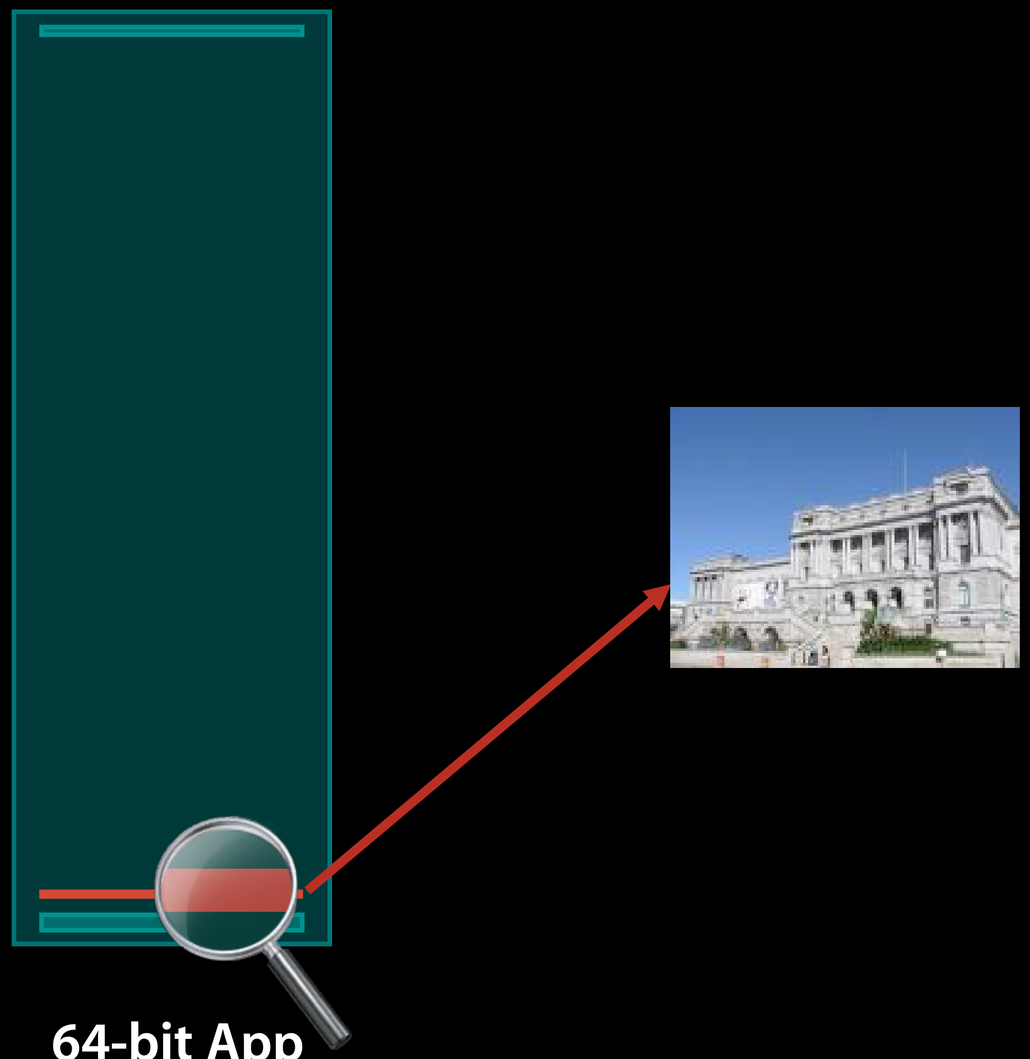
In Internet Archives

64-bit App
(*not to scale)

X ≈16 thousand

How Gigantic Is a 64-bit Address Space?

In Libraries of Congress

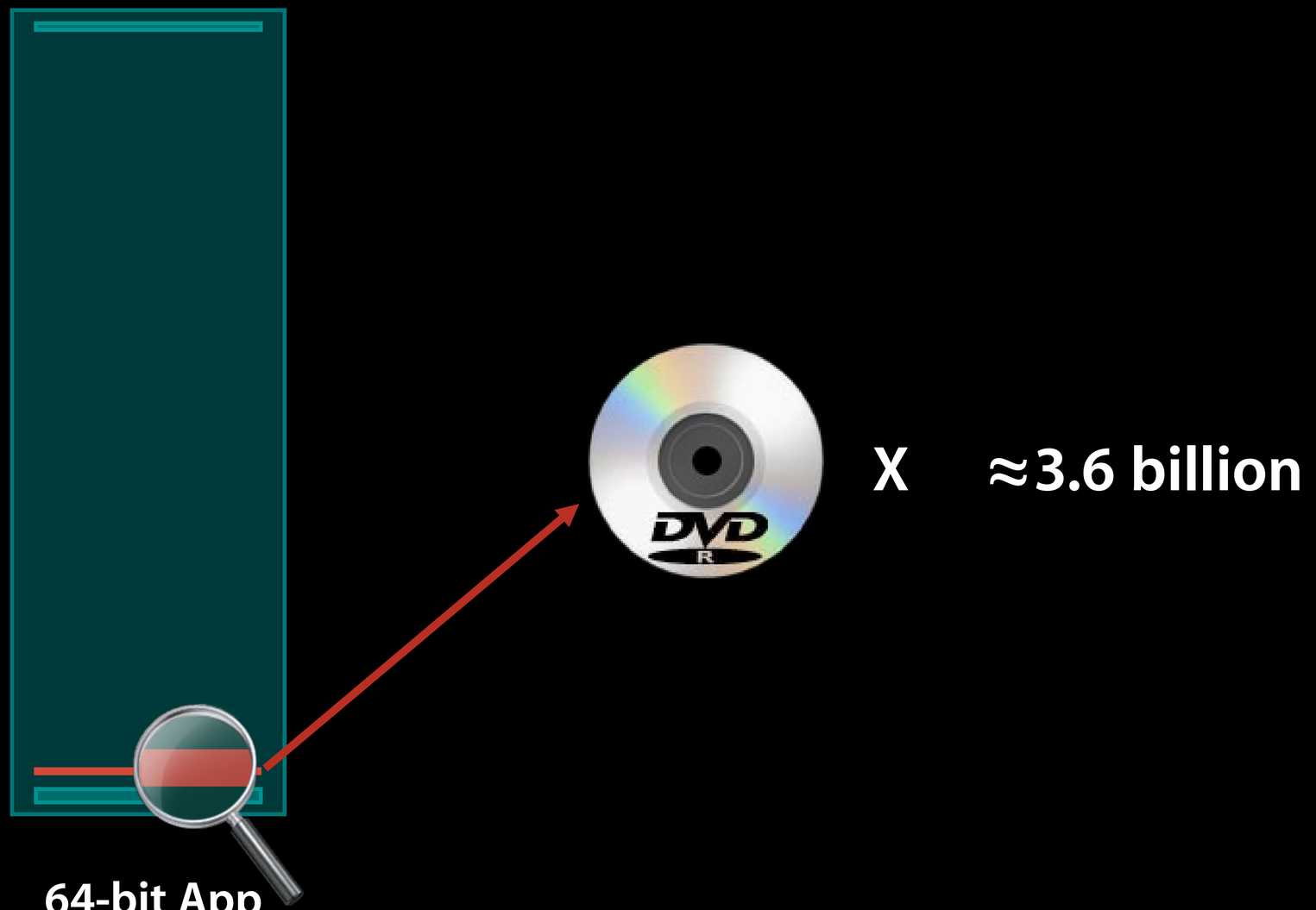


X ≈1 million

64-bit App
(*__STILL__ not to scale)

How Gigantic Is a 64-bit Address Space?

in DVDs (single layer)

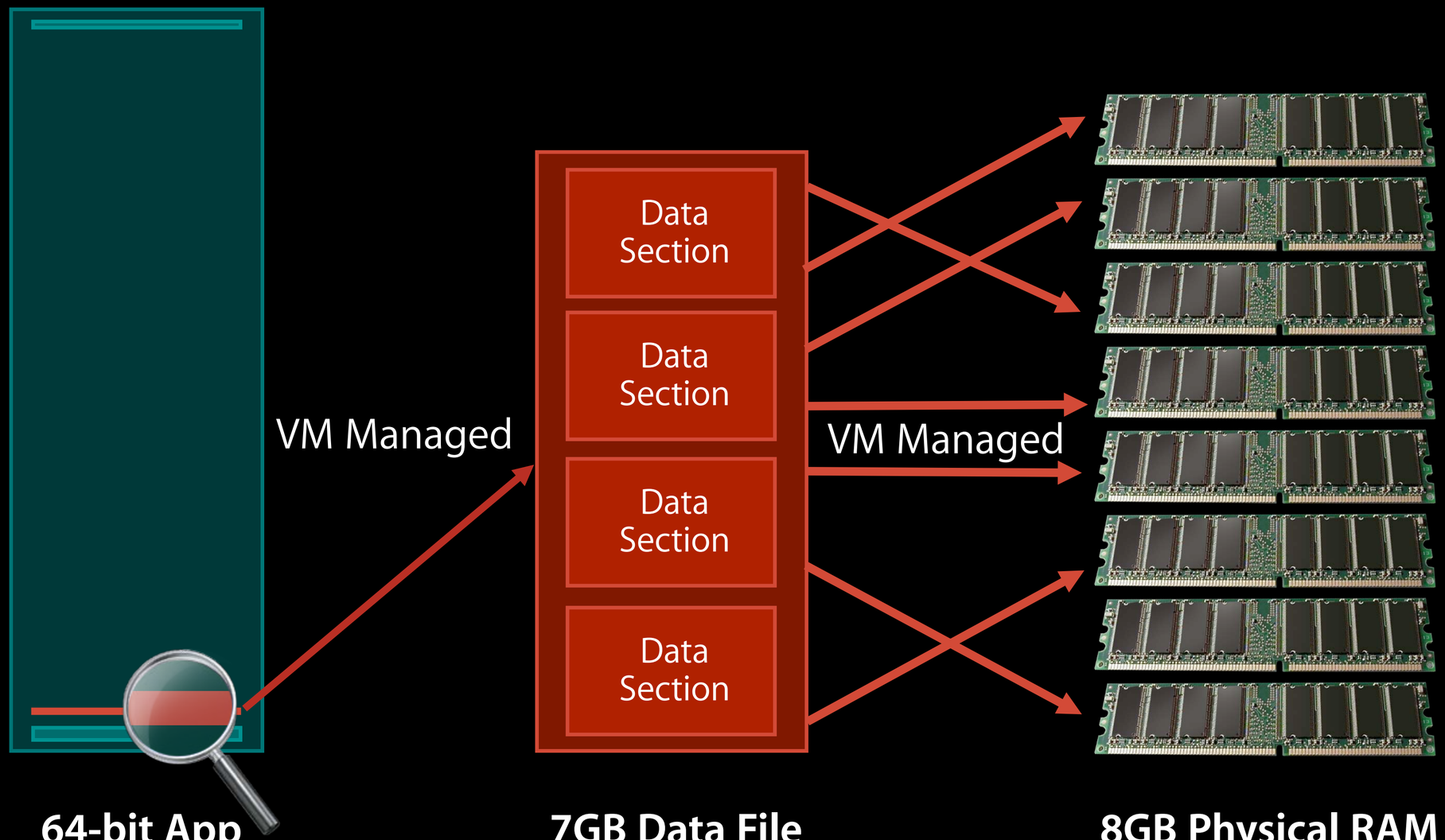


X ≈3.6 billion

64-bit App
(*__STILL__ not to scale)

Large Data Sets in 64-bit Applications

Can easily map it all with a *bit* of space left over



The Future of 64-bit in Mac OS X

- Additional Apple 64-bit extensions in stages
  - Timing and specifics decided with feedback from developer community
- Open source world adds at own schedule
  - e.g. X11 client libs

When to Develop 64-bit Applications

http://developer.apple.com/wwdc2005/

64-bit vs. 32-bit Tradeoffs

- Compute-intensive, long or long long integers
  - WIN

64-bit vs. 32-bit Tradeoffs

- Compute-intensive, long or long long integers
  - WIN
- Pointer-intensive, small amounts of data in memory
  - ~~LOSE~~ WIN LESS

64-bit vs. 32-bit Tradeoffs

- Compute-intensive, long or long long integers
  - WIN
- Pointer-intensive, small amounts of data in memory
  - ~~LOSE~~ WIN LESS
- Very large amounts of data in memory
  - WIN

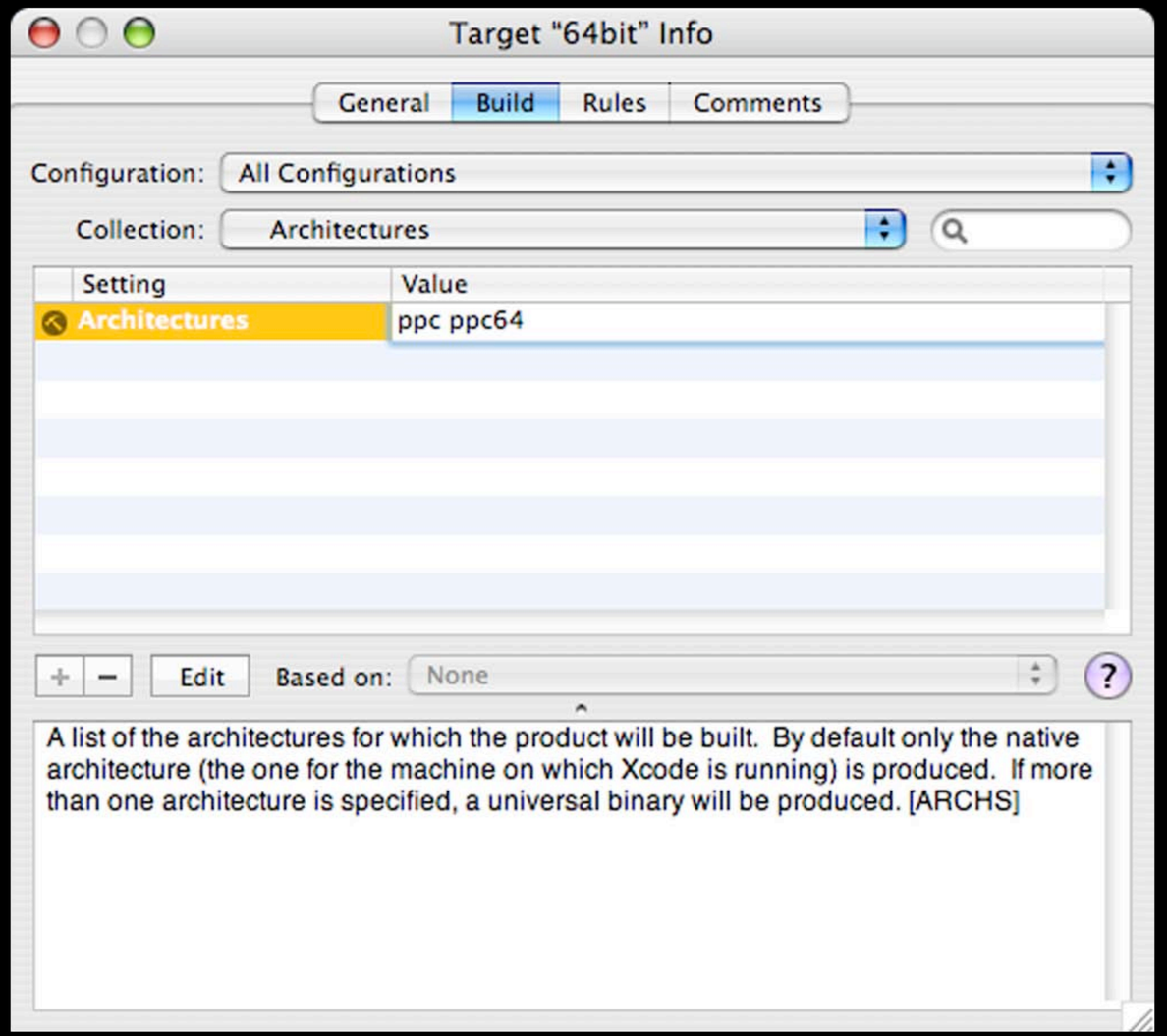
64-bit vs. 32-bit Tradeoffs

- Compute-intensive, long or long long integers
  - WIN
- Pointer-intensive, small amounts of data in memory
  - ~~LOSE~~ WIN LESS
- Very large amounts of data in memory
  - WIN
- 64-bit Windows
  - LOSING CONTINUOUSLY

How to Develop 64-bit Applications

http://developer.apple.com/wwdc2005/

How to Get 64-bit Code



64-bit Programming Considerations

http://developer.apple.com/wwdc2005/

64-bit Data Model Is LP64

	ILP32	LP64	ILP64	(LL)P64
int	32-bit	32-bit	64-bit	32-bit
long	32-bit	64-bit	64-bit	32-bit
long long	64-bit	64-bit	64-bit	64-bit
pointer	32-bit	64-bit	64-bit	64-bit
Platforms	—	Tiger, Unix, Sun and SGI	Alpha, Cray	Windows

Programs In the LP64 World

- Some source code changes may be required
- New practices for 64-bit
  - ints cannot hold pointers
  - Integral constant suffixes (`L, UL, LL, ULL`)
  - printf directives (`%ld, %lld`)
- Things to watch out for
  - Casting can destroy your data
  - Sign extensions can confuse your data
  - Function calls can lose your data
  - 64-bit to 32-bit conversion can scramble your data
  - Alignment rules can “hole” your data

ints and Pointers

- Assignment of pointers to ints LOSES DATA
- Assignment to/from long or long long is okay

```
int intvar;
long longvar;
char *pointervar;

pointervar = malloc(0x800000000);
  // pointervar == 0x123456780000
intvar = pointervar;  /* “assignment makes integer
from pointer without a cast” */
  // intvar == 0x56780000 (!)
longvar = pointervar;
  // longvar == 0x123456780000
```

Casting

- Casting is not magic
- May suppress critical warnings

```
int ivar;
long lvar;
char *pvar = (char *) 0x12345678901;

ivar = (int) pvar;
  // ivar == 0x45678901
lvar = (long) pvar;
  // lvar == 0x12345678901
```

Sign Extensions

An unsigned 64-bit number may look like a signed 32-bit

```
unsigned int uintvar = 0x90000000;
unsigned long ulongvar = (unsigned long)
uintvar;
int intvar = (int) ulongvar; // intvar < 0
long longvar = (long) intvar; // longvar < 0

ulongvar = (unsigned long) intvar;
  // ulongvar == 18446744071830503424 (!)
  // ulongvar == 0xffffffff90000000
```

Function Calls

- Correct prototypes are critical
- Ask compiler to warn about prototypes

```
int fun();

int main() {
 long xx = 0x12345678abcdef0;
 fun(xx);
 fun(0x12345678abcdef0);
}

int fun (int x) {
  return x;  // x == 0x8abcdef0
}
```

64 to 32-bit Extraction

Use masking or unions to extract the two 32-bit values

```
union splitter { long big; int smalls[2] } xx;

long longvar = 0x123456789012;

xx.long = longvar;

printf(”%x %x\n”, xx.smalls[0], xx.smalls[1]);
1234 56789012

int intlo, inthi;

intlo = longvar & (0x00000000ffffffffL);
inthi = longvar >> 32;

printf(”%x %x\n”, inthi, intho);
1234 56789012
```

Writing Compatible Code

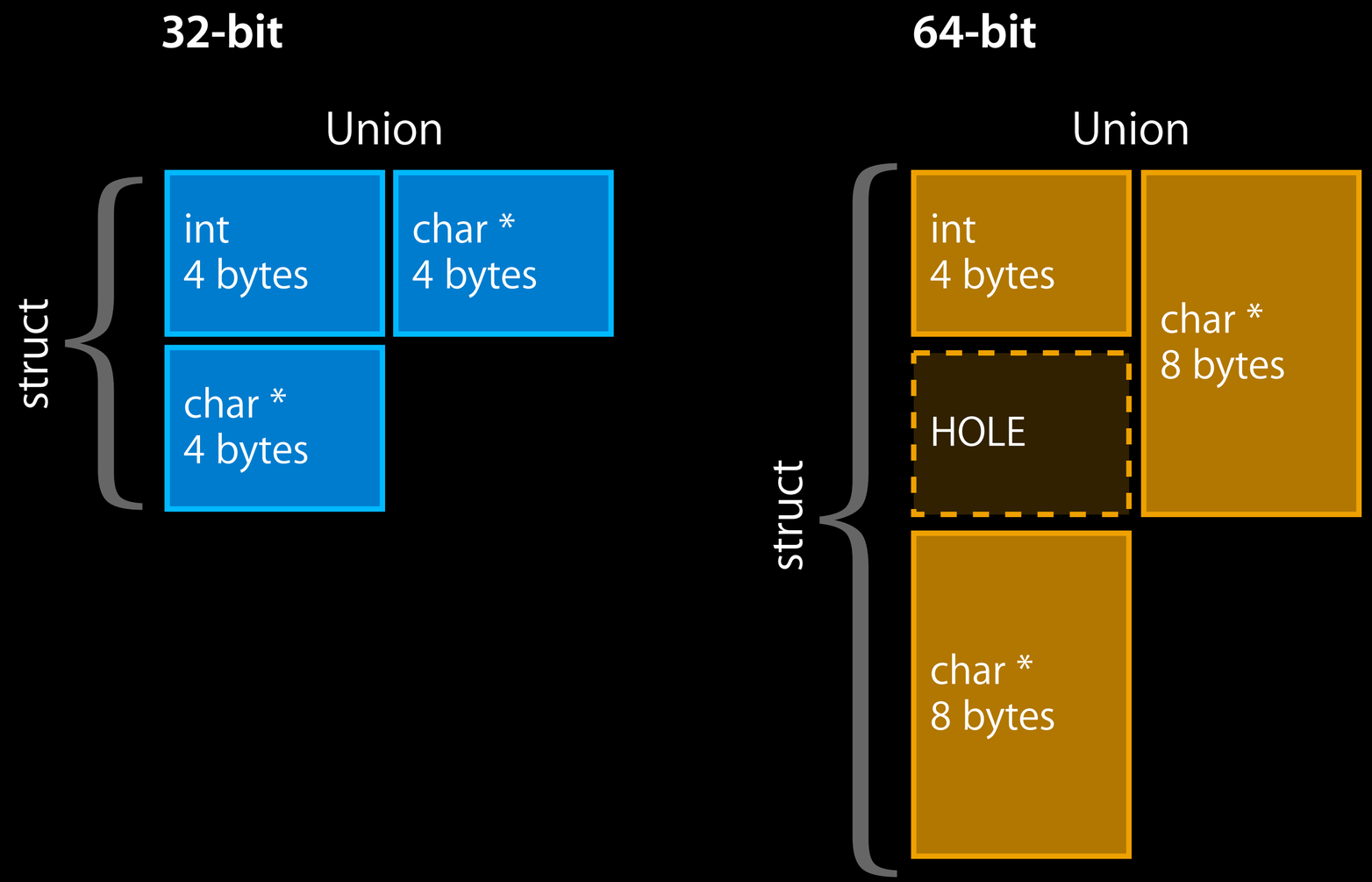
- C90 / C99 standard types and macros
  - `size_t, intptr_t, uintptr_t`

- Unix standard types
  - `uint32_t, int64_t` ...

- Locally defined standard types
  - Many programs have these already
  - printf macros
    - `#define LL_PRINT_DEC "%lld"`
    - `printf("value=" LL_PRINT_DEC "\n",  longlongvar);`

Alignment Trickery

Alignment rules can create unexpected holes in data structures



API Changes

- API support macros
  - `__LP64__`
  - `__ppc64__` (vs. `__ppc__`)

- In prototypes, long --> `int if __LP64__`

```
#ifdef __LP64__
 int  getattrlist(const char*,void*,void*,size_t, unsigned int);
#else /*__LP64__*/
 int  getattrlist(const char*,void*,void*,size_t, unsigned long);
#endif
```

- .quad in assembly language
  - `.quad 0x123456789012`

Tiger 64-bit Considerations

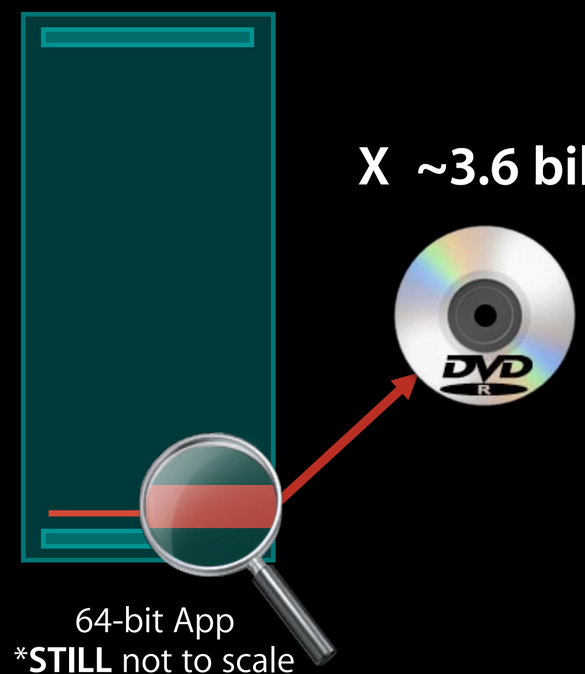
http://developer.apple.com/wwdc2005/

Mac OS X Tiger 64-bit Goals/Limitations

- Designed to meet the needs of many typical 64-bit programs
  - Servers, databases, compute engines
  - C and C++
  - Command line
  - Tools launched by plugins
- Only the necessary 64-bit libraries provided
  - libSystem
  - Accelerate framework (vecLib)

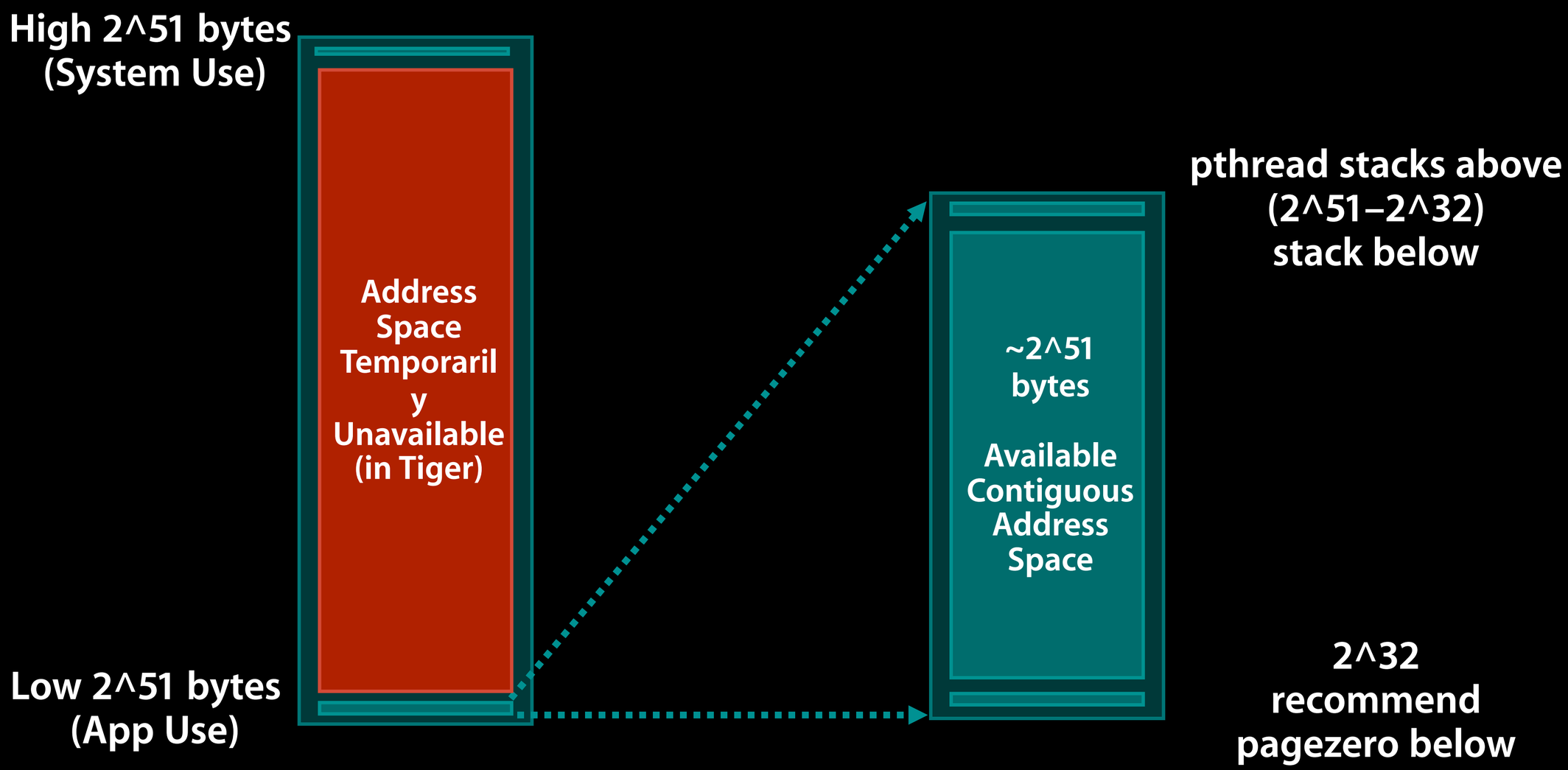
Just Because You Could Map 3 Billion DVDs…

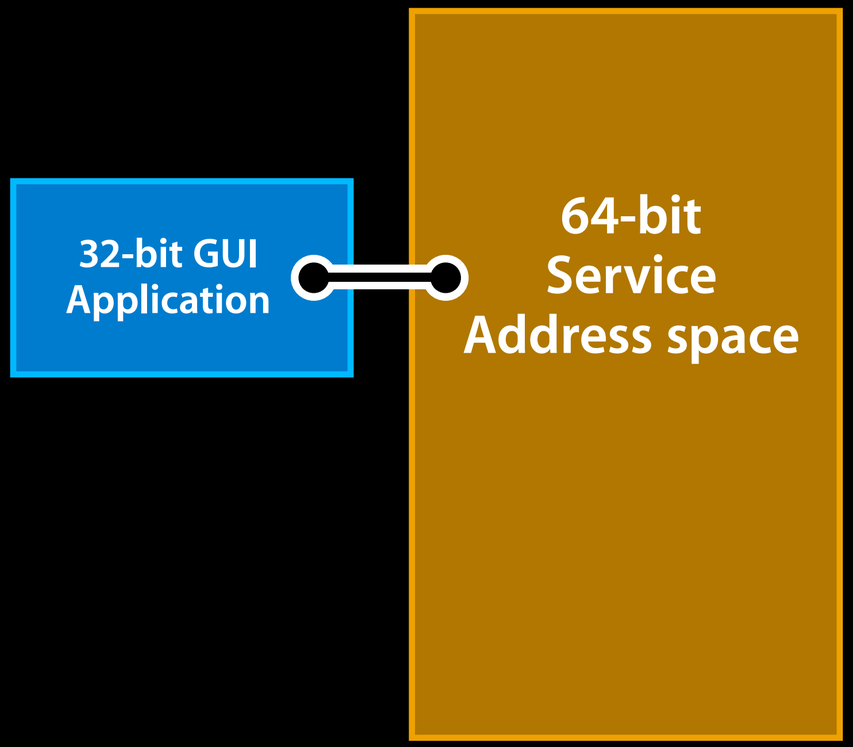
…doesn't mean you should



- Tiger designed for thousands of mappings per address space
  - Not billions
- Scaled for giving "convenient" access to all of physical memory
  - As opposed to mapping the Internet Archive
- Besides…
  - Not all 64-bits available right now

Tiger's 64-bit Address Space Layout





There Are No 64-bit GUI Frameworks in Tiger

You were wondering when we were going to get to this—admit it

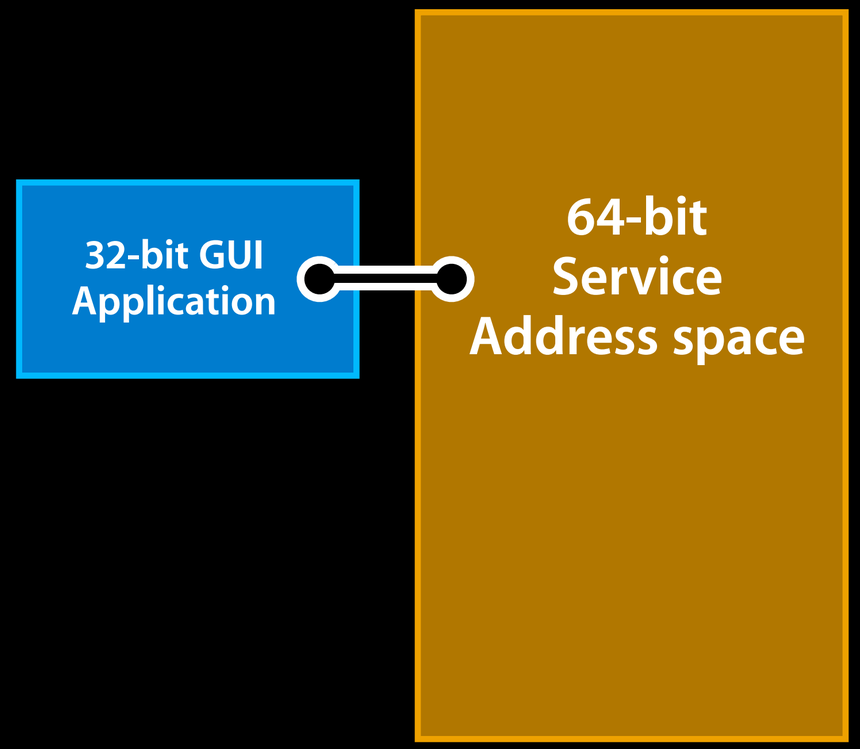
- Most applications in target categories don't have integrated GUIs, so it's not an issue
- Those that do can communicate with a 32-bit GUI front-end using Inter-Process Communication (IPC) mechanisms

IPC for 64-bit Processes

Jim Magee
Software Engineer

http://developer.apple.com/wwdc2005/

“It’s just a little black line, how difficult can it be?”

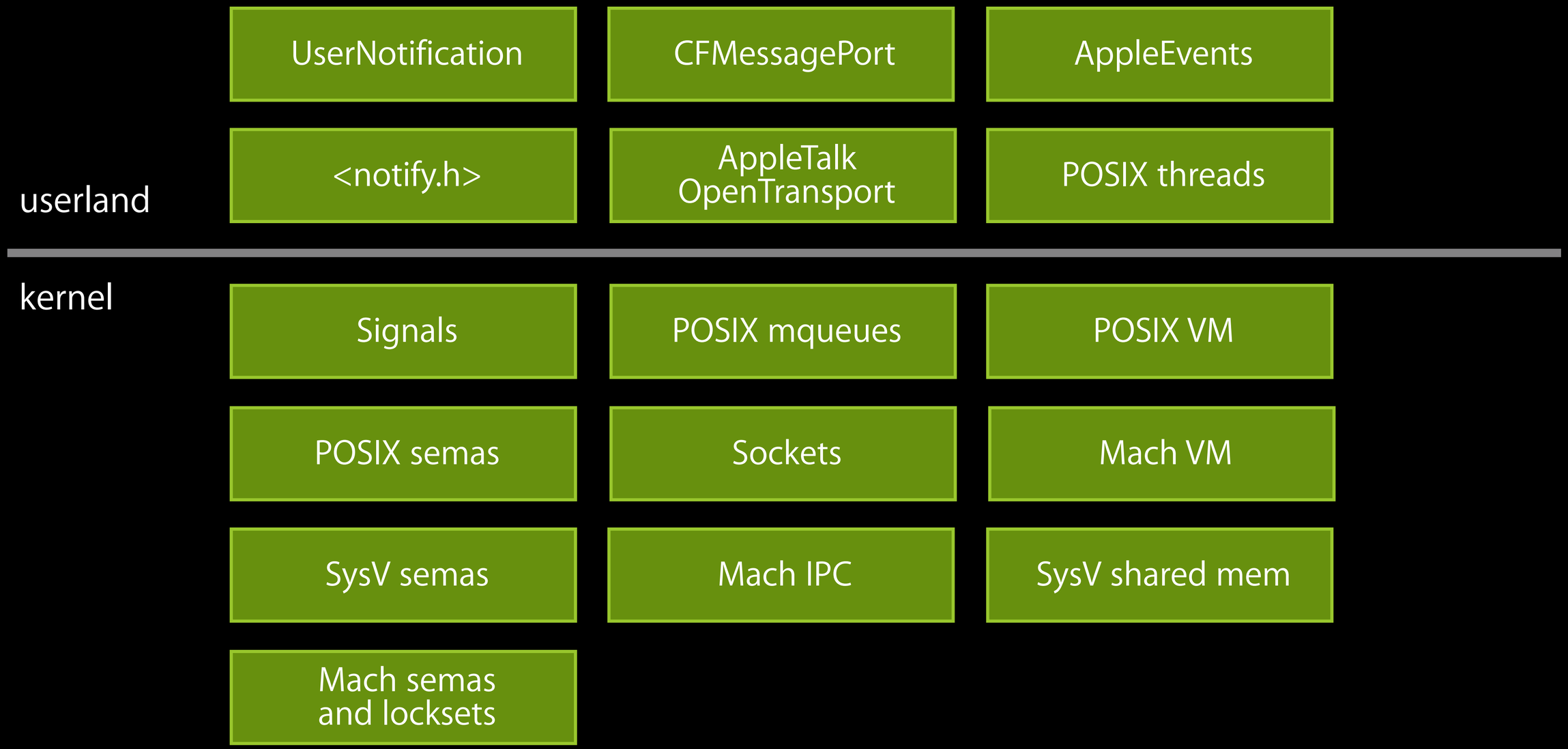


A Software VP at a former employer (who shall not be identified)

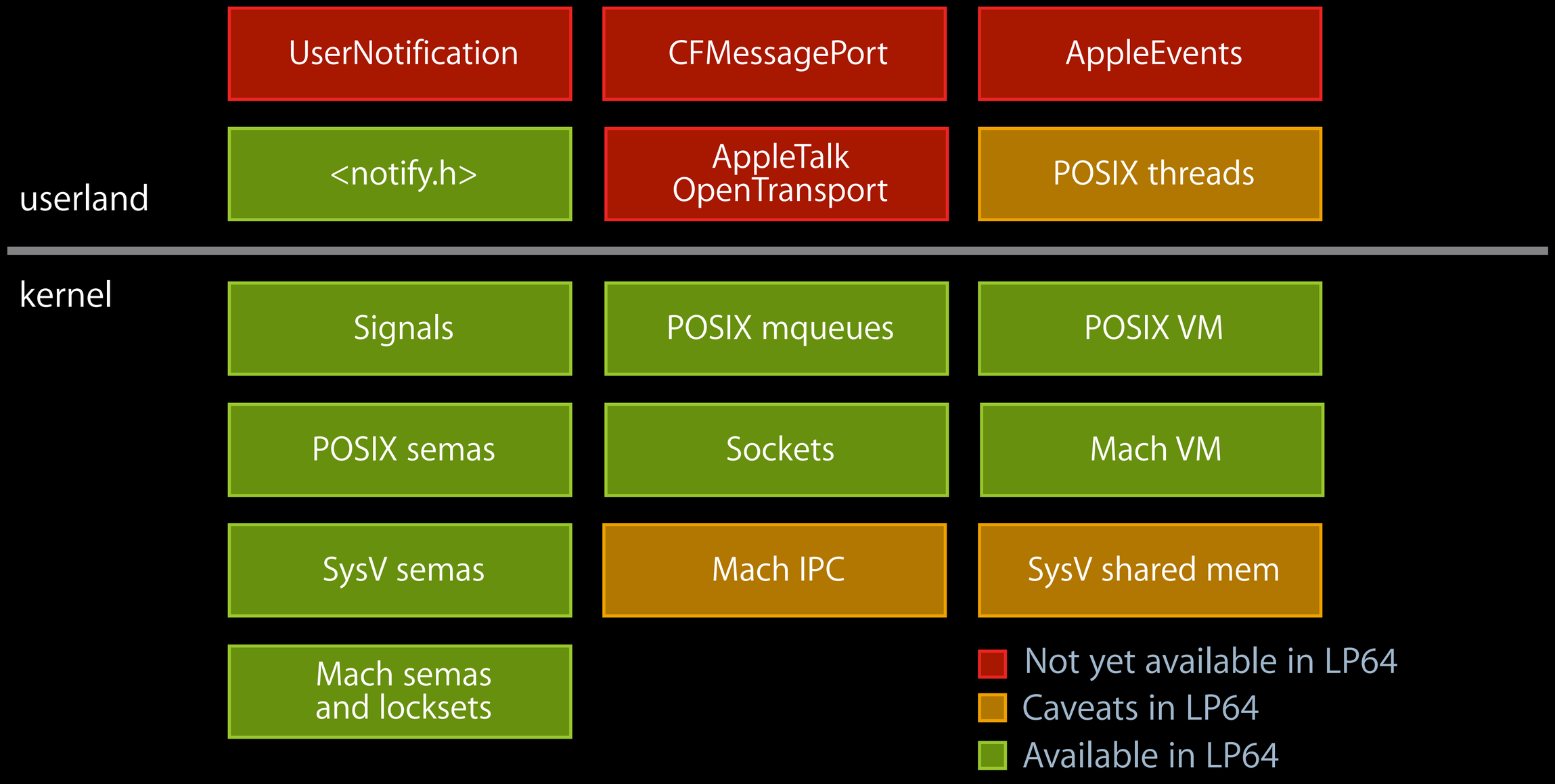
IPC in 64-bit Processes

- Enable 32-bit GUI front-end and 64-bit back-end
  - Make it as simple as possible
- Support IPC between 64-bit and other 32-bit processes
  - System daemons/services
  - Largely hidden by the 64-bit frameworks
- Also support 64-bit process to 64-bit process IPC

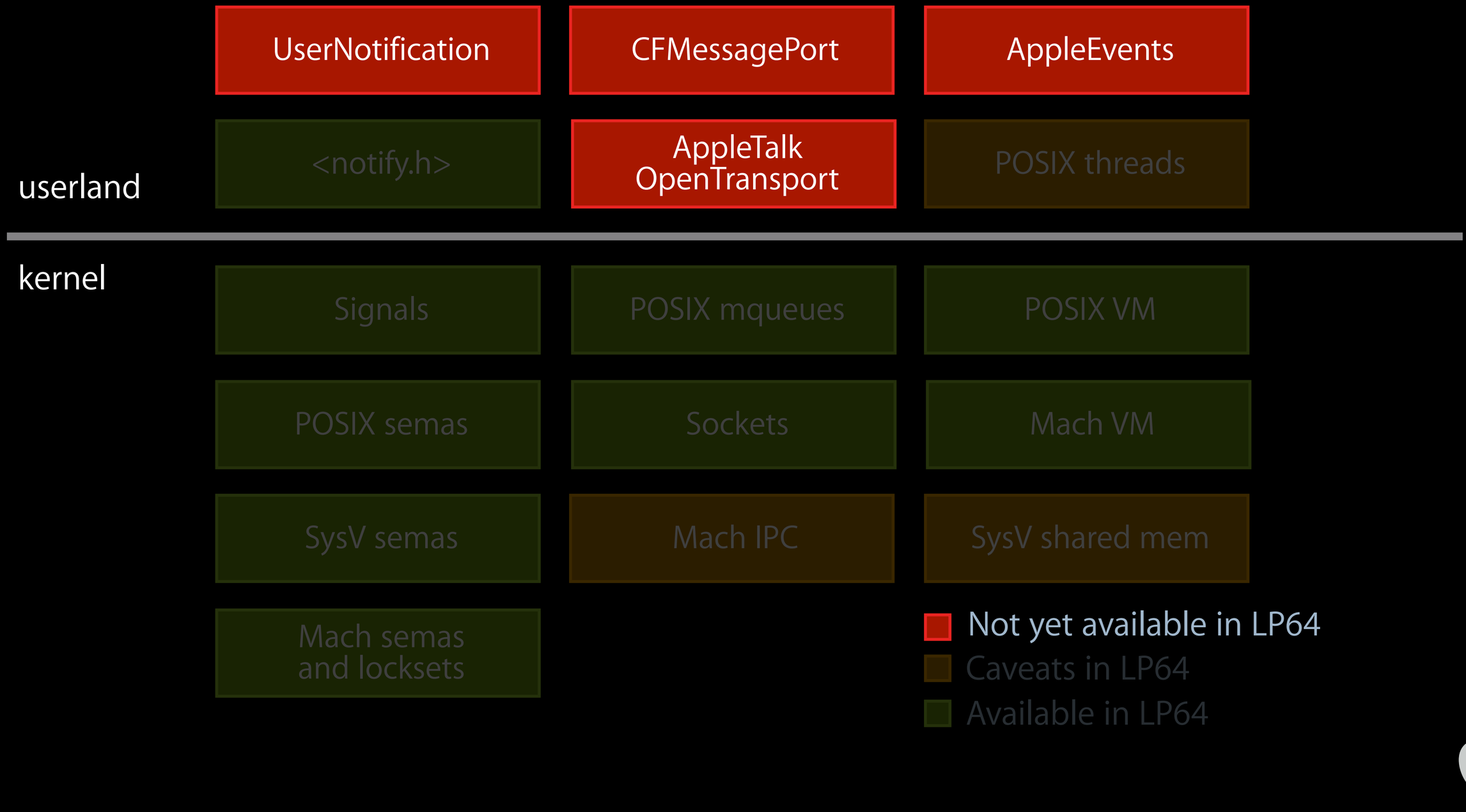
IPC Services in Mac OS X



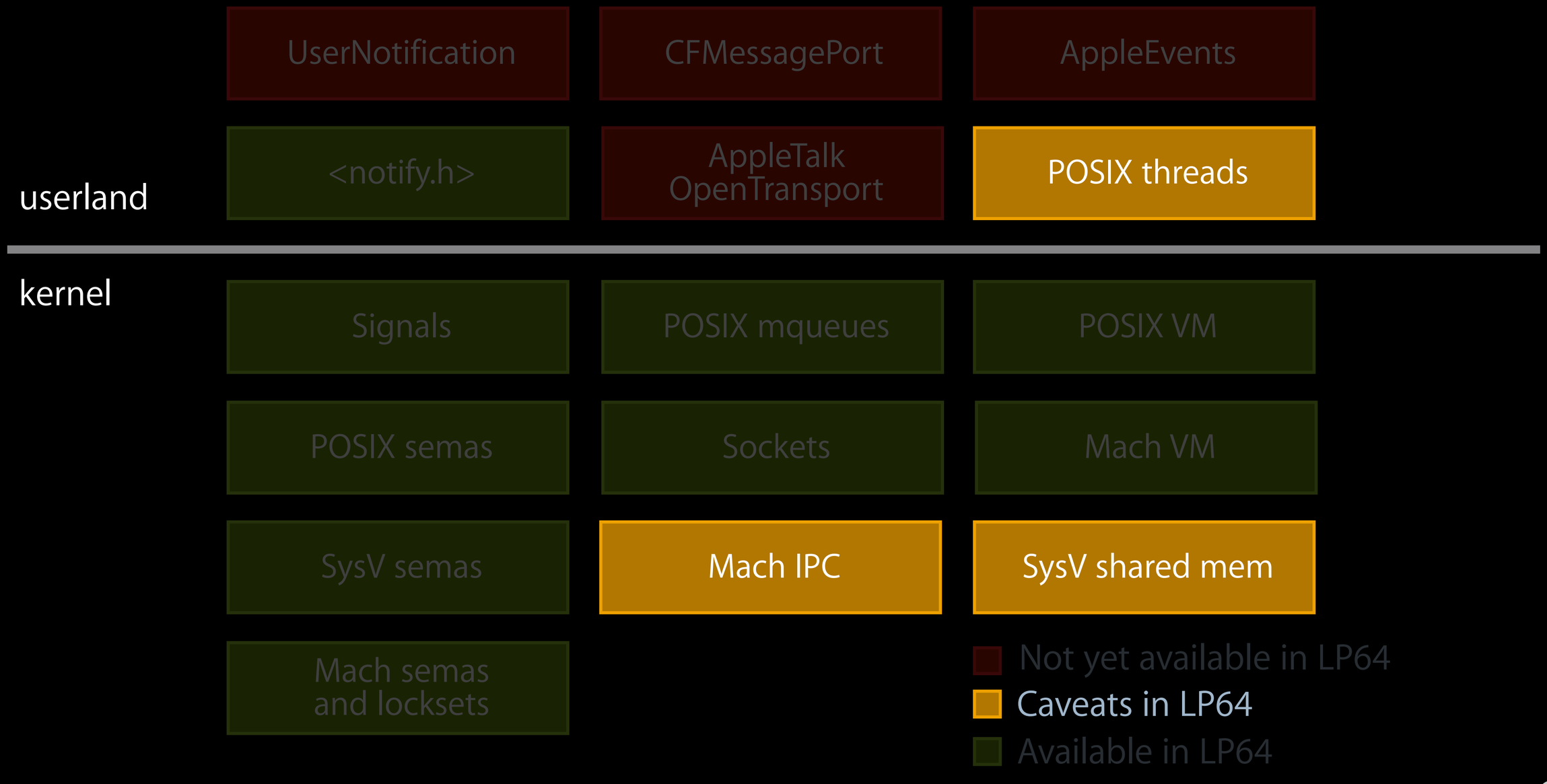
IPC Service Support in LP64



IPC Service Support in LP64



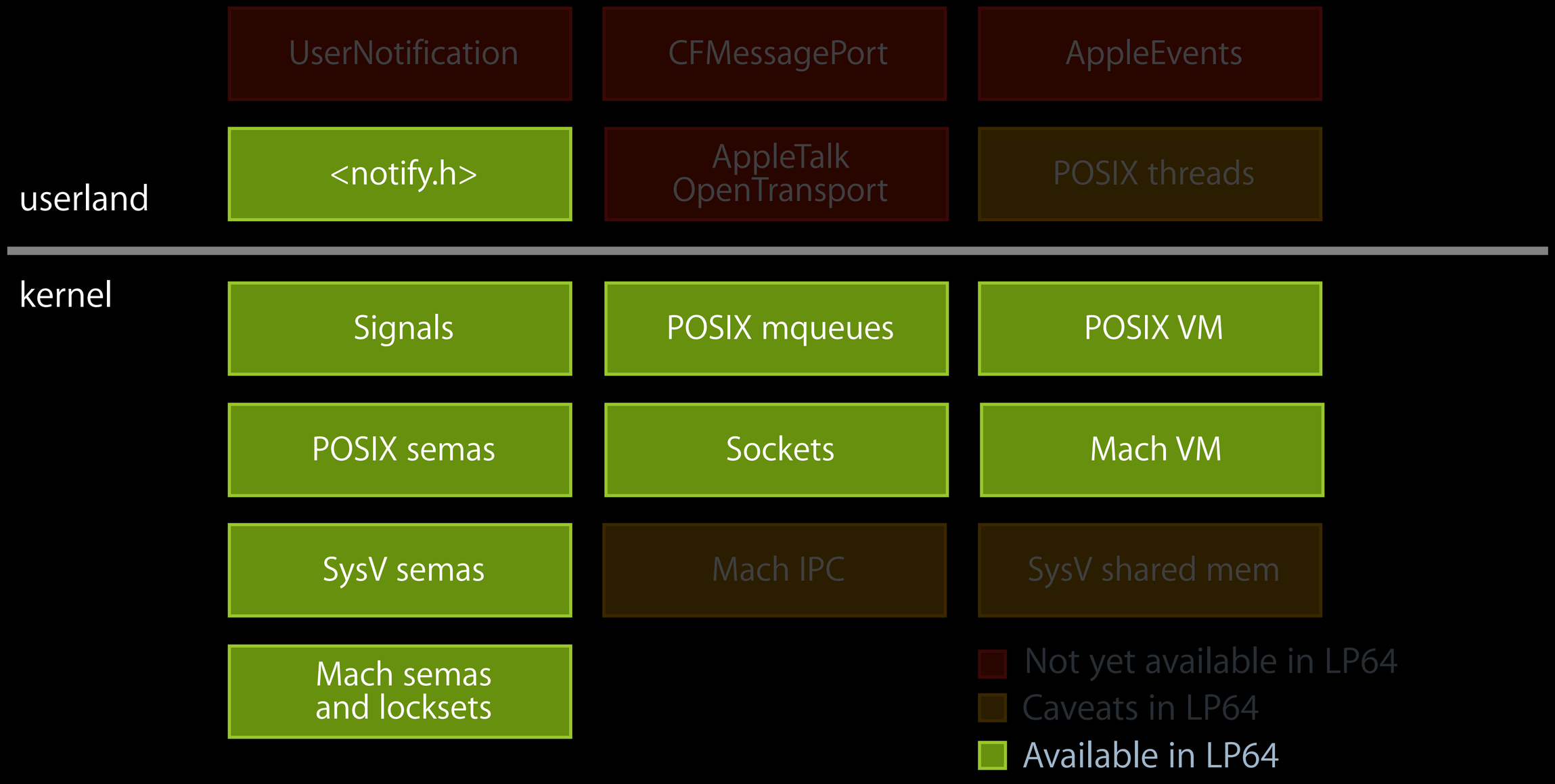
IPC Service Support in LP64



IPC Services With Caveats in LP64

- POSIX thread synchronizers
  - PSHARED attribute not currently supported
    - So can’t be used to coordinate GUI and LP64 back-end processes
- System V shared memory
  - Limited maximum size of regions
    - Much smaller than what LP64 processes can share otherwise
- Mach IPC
  - Individual out of line regions “limited” to 4GB length each
    - Can send multiple to compensate
    - Or send a memory handle as a port

IPC Service Support in LP64



GUI32-tool64 IPC Recommendations

- Construct as you would any tool-using application
  - GUI application with the tool as a “resource”
  - Take advantage of Unix/POSIX layer in Mac OS X
- Continue to use the highest-level IPC service
  - That fits the need
  - Available to both 32-bit and 64-bit processes
- Can sometimes combine IPC services
  - RunLoop compatible in 32-bit GUI app
  - Raw/native facility in 64-bit tool
  - e.g. CFSocket in the GUI app; raw socket/pipe in 64-bit tool

GUI32-tool64 IPC Recommendations (Cont.)

- Take advantage of parent/child relationships
  - 32-bit GUI app launch 64-bit tool
  - Use inheritance
    - POSIX file descriptors
      - stdin, stdout, stderr pipes/sockets
      - For mmap(MAP_SHARED)
      - POSIX semaphores
      - POSIX shared memory
    - Mapped memory (through minherit())
- But must “manage” the running tool
  - Use IPC invalidation callbacks to trigger if possible
  - sigchild is also your friend

GUI32-tool64 IPC Example Code

Stdin/Stdout for IPC from tool64.c

```
#include <stdio.h>

/* "scan input from stdin" */
size_t count;
int flag = scanf("%zi", &count);
if (EOF == flag) break;

/* "print ouput to stdout" */
void* ptr1 = malloc(count);
printf("ptr1 == %p!\n", ptr1);
fflush(stdout);
```

GUI32-tool64 IPC Example Code

Launch tool64 and hook up stdin/stout from GUI32.app

```
/* "get tool path and launch it" */
char path[PATH_MAX];
toolCFURLRef = CFBundleCopyAuxiliaryExecutableURL(...);
CFURLGetFilesystemRepresentation(toolCFURLRef,...,path,...);
gPipeFile = popen(path, "r+");

/* "install results handling callback" */
if (gPipeFile) {
  setlinebuf(gPipeFile);
  Install_Callback(gPipeFile, myCFSocketCallback);
}
```

GUI32-tool64 IPC Example Code

Hook tool's output socket/pipe into CFRunLoop in GUI32.app

```
Install_Callback(FILE *inPipe, CFSocketCallBack inCallback)
{
/* "get pipe fd and create a CFSocket around it" */
int fdNum = fileno(inPipe);
CFSocketRef cfSock = CFSocketCreateWithNative(NULL,fdNum,
                                              inCallback,...);

/* "create a RunLoop source for it and add it to the RunLoop" */
CFRunLoopSource rls = CFSocketCreateRunLoopSource(NULL,cfSock,...);
CFRunLoopAddSource(CFRunLoopGetCurrent(),rls,...);
}
```

Demo

Using a 64-bit tool from a 32-bit GUI application

http://developer.apple.com/wwdc2005/

Wolfram Research
64-bit Mathematica on Mac OS X

Rob Raguet-Schofield
Software Engineer
Wolfram Research, Inc.
schofield@wolfram.com

http://developer.apple.com/wwdc2005/

Technology Adoption to Add Value

altivec

Aqua UI

64 Bit

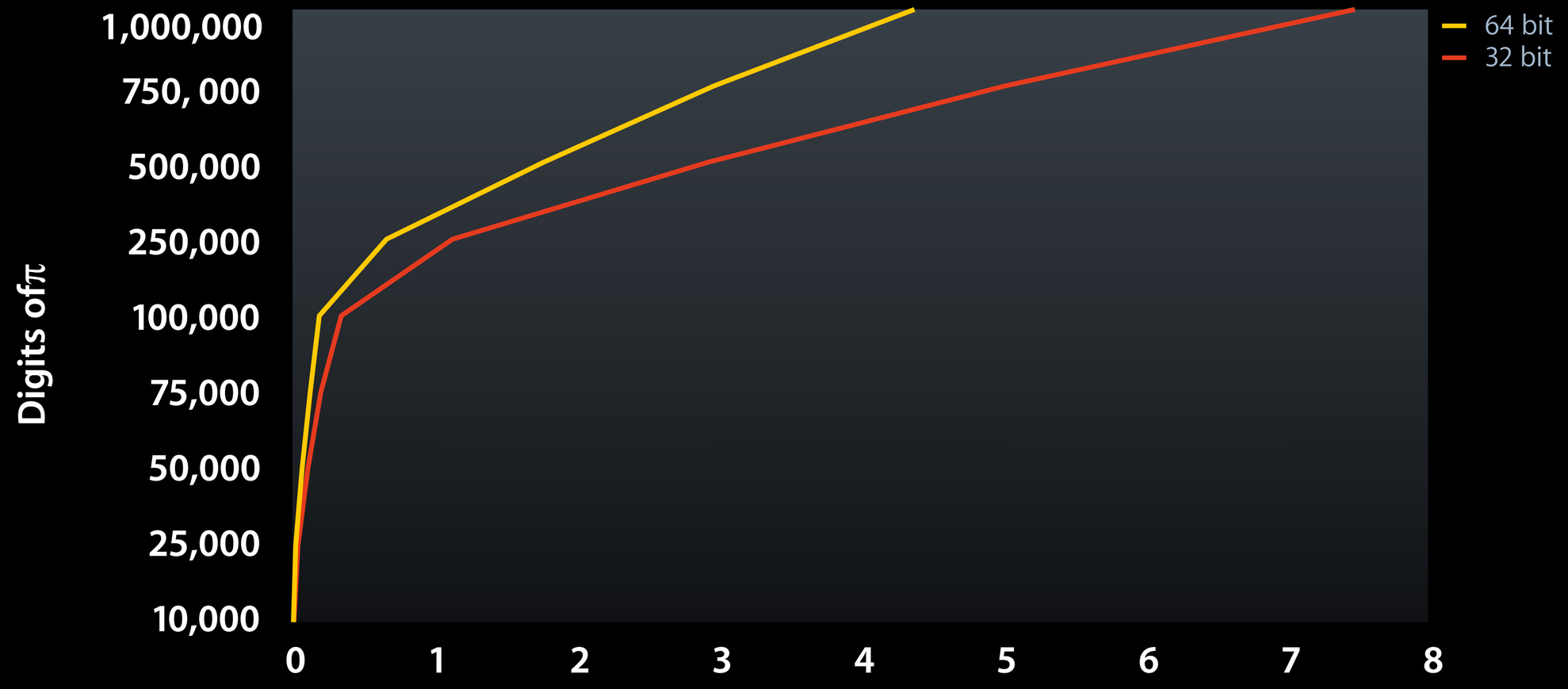
Java

OpenGL

Spotlight

Automator

64-bit vs. 32-bit Arbitrary Precision Mathematics



Demo

Mathematica 5.2 on Mac OS X Tiger

http://developer.apple.com/wwdc2005/

More Information

http://developer.apple.com/wwdc2005/

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Related Sessions

64-bit Development Lab	Lab Wednesday 3:30PM
Core OS Enhancements for BSD Developers	Session 510 Thursday 3:30PM
Swimming with Shark	Session 407 Friday 2:00PM
Feedback Forum: Development Tools	Forum Friday 5:00PM

Who to Contact

Matthew Formica
64-bit Evangelist
mformica@apple.com

Q&A

Matthew Formica
64-bit Evangelist
mformica@apple.com

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference